CPSC 468 Midterm Review

Chapter 1: The Computational Model and Why It Doesn't Matter

0.1 (Turing Machine)**.** A k-tape Turing Machine M is described by a tuple (Γ, Q, δ). Assume $k \geq 2$, with 1 read-only input tape and $k-1$ work tapes. The last work tape is assumed to be the output tape.

- Γ the finite alphabet of symbols that M may have on its tapes. Assume that Γ contains at least $\{0, 1, \square, \triangleright\}$.
- Q a finite set of possible states M's state register may be in. Assume that Q contains a q_{start} and q_{halt}.
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^{k-1} \times \{L, S, R\}^k$ a transition function for M that takes in the current state and each head's read, and outputs the next state, with $k-1$ writes on all the work tapes, and movements for all k tapes.

0.2 (Computing a Function)**.** Let $f : \{0,1\}^* \to \{0,1\}^*$ and $T : \mathbb{N} \to \mathbb{N}$, with M a TM. We say that M computes f if for every $x \in \{0,1\}^*$, if M is initialized to the start configuration on input x, then it halts with $f(x)$ on the output tape. We say M computes f in $T(n)$-time if its computation on every x requires at most $T(|x|)$ steps.

0.3 (Time Constructible)**.** A function $T : \mathbb{N} \to \mathbb{N}$ is time constructible if $T(n) \geq n$ and there is a TM M that computes the function $x \mapsto \llcorner T(|x|) \lrcorner$ in time $T(n)$. $T(n) \geq n$ is to allow the algorithm to read its input. Some time constructible functions are n, $n \log n$, n^2 and 2^n.

0.4. For every $f : \{0,1\}^* \to \{0,1\}$ and time constructible $T : \mathbb{N} \to \mathbb{N}$, if f is computable in time $T(n)$ by some TM M using alphabet Γ, then it is able to compute the same function using $\{0, 1, \square, \triangleright\}$ in $(c \log_2 |\Gamma|) \cdot T(n)$. This is because we may express each symbol of Γ using $\log |\Gamma|$ binary bits, with some constant c overhead.

0.5. A k-tape TM can have its $k-1$ work tapes simulated by a single tape by interleaving the k tapes together.

0.6 (Oblivious Turing Machine)**.** An oblivious TM's head movement depends on the length of the input, not the contents of the input. Every TM can be simulated by an oblivious TM.

0.7 (Turing Machine Representation)**.** Every binary string $x \in \{0,1\}^*$ represents some TM, and every TM is represented by infinite such strings (think: comments in a language). The machine represented by x is denoted M_x.

0.8 (Universal Turing Machine)**.** There exists a TM $\mathcal{U}$ such that for every $x, \alpha \in \{0,1\}^*$, $\mathcal{U}(x, a) = M_\alpha(x)$, where M_α denotes the TM represented by α. Moreover, if M_α halts on input x within T steps, then $\mathcal{U}_\alpha(x)$ halts within $CT \log T$ steps, where C is a number independent of $|x|$, and depends only on M_α's alphabet size, number of tapes, and number of states. The cost of simulating any machine M_α has a logarithmic overhead, due to the alphabet size difference between M_α and $\mathcal{U}$. As $\mathcal{U}$ has a single tape, we do the trick over interleaving M_α's work tapes together.

0.9 (Uncomputable Function)**.** Define U as follows: for every $\alpha \in \{0,1\}^*$, if the machine defined by α accepts itself, such that $M_\alpha(\alpha) = 1$, then $U(\alpha) = 0$. In other words $U(\alpha) = 1 - M_\alpha(\alpha)$. There is no such TM that can compute U, because U will always negate it.

0.10 (Halting Problem)**.** A TM H such that $H(\alpha, x) = 1$ if $M_\alpha(x)$ halts, and yields 0 otherwise, does not exist. We can construct a wrapper TM W that invokes H on itself, and performs the opposite. Diagonalization motherfuckers!

0.11 (DTIME)**.** Let $T : \mathbb{N} \to \mathbb{N}$ be some function. A language L is in DTIME($T(n)$) iff there is a deterministic TM that runs in time $c \cdot T(n)$ for some constant $c > 0$ and decides L. This class contains **decision** problems.

0.12 (The Class P)**.** $\mathrm{P} = \bigcup_{c \geq 1} \mathrm{DTIME}(n^c)$

0.13 (Church-Turing Thesis)**.** Every physically realizable computation device can be simulated by a TM.

0.14 (Bounds)**.** The asymptoptic operators $\{o, O, \Theta, \Omega, \omega\}$ can be thought of as $\{<, \leq, =, \geq, >\}$.

Chapter 2: NP and NP-completeness

0.15 (Non-deterministic Turing Machine)**.** A non-deterministic TM is endowed with two transition functions δ_0 and δ_1 along with a special accept state q_{accept}. The NDTM may use either transition function per time step. For every input x, we say that $M(x) = 1$ if there **exists** some sequence of transition function choises that would cause M to reach q_{accept}. Otherwise, if every sequence of non-deterministic choices causes M to halt on x without reaching q_{accept}, then we say that $M(x) = 0$. M runs in time $T(n)$ if for every input $x \in \{0,1\}^*$ and every sequence of non-deterministic choices, M reaches eitehr the halting state or q_{accept} within $T(|x|)$ steps.

0.16 (NTIME)**.** For every function $T : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$, we say that $L \in \mathrm{NTIME}(T(n))$ if there is a constant $c > 0$ and a $c \cdot T(n)$-time NDTM M such that for every $x \in \{0,1\}^*$, we have $x \in L \iff M(x) = 1$.

0.17 (NP)**.** $\mathrm{NP} = \bigcup_{c\in\mathbb{N}} \mathrm{NTIME}(n^c)$

0.18. A language $L \subseteq \{0,1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial time TM M (called the **verifier** for L) s.t. for every $x \in \{0,1\}^*$, we have $x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$. If $x \in L$ and $u \in \{0,1\}^{p(|x|)}$ satisfy $M(x,u) = 1$, we call u a **certificate** for x (with respect to L and M). NP is the class of languages for which we can tell if u is a solution to the problem $x \in L$ in polynomial time.

0.19 (Reductions, NP-hardness, and NP-completeness)**.** We say that a language $L \subseteq \{0,1\}^*$ is a polynomial time **Karp reducible** to a language $L' \subseteq \{0,1\}^*$ if there is a polynomial time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for every $x \in \{0,1\}^*$, we have $x \in L \iff f(x) \in L'$. We say that L' is NP-hard if $L \leq_p L'$ for every $L \in \mathrm{NP}$. We say that L' is NP-complete if L' is NP-hard and $L' \in \mathrm{NP}$.

0.20 (Cook Reduction)**.** A reduction computed by a deterministic polynomial time oracle TM.

0.21 (Transitivity)**.** If $L \leq_p L'$, and $L' \leq_p L''$, then $L \leq_p L''$.

0.22. If L is NP-hard and $L \in \mathrm{P}$, then $\mathrm{P} = \mathrm{NP}$.

0.23. If L is NP complete then $L \in \mathrm{P} \iff \mathrm{P} = \mathrm{NP}$.

0.24 (TMSAT)**.** $\mathrm{TMSAT} = \{(\alpha, x, 1^n, 1^t) : \exists u \in \{0,1\}^n \text{ s.t. } M_a(x,u) = 1 \text{ within } t \text{ steps}\}$

0.25 (Decision vs Search)**.** Suppose that $\mathrm{P} = \mathrm{NP}$, then for every NP language L there exists a polynomial time TM B such that on input $x \in L$ outputs a certificate for x. That is, $x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$ s.t. $M(x,u) = 1$ where p is some polynomial and M is a polynomial time TM, then on input $x \in L$, we have $B(x)$ as the string $u \in \{0,1\}^{p(|x|)}$ satisfying $M(x, B(x)) = 1$.

0.26 (Complement Language)**.** If $L \subseteq \{0,1\}^*$, then we denote $\overline{L}$ to be the complement of L. That is, $\overline{L} = \{0,1\}^* \setminus L$.

0.27 (coNP)**.** $\mathrm{coNP} = \{L : \overline{L} \in \mathrm{NP}\}$

0.28 (coNP Alternative Definition)**.** For every $L \subseteq \{0,1\}^*$, we say $L \in \mathrm{coNP}$ if there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a polynomial time TM M s.t. for every $x \in \{0,1\}^*$, we have $x \in L \iff \forall u \in \{0,1\}^{p(|x|)}$ with $M(x,u) = 1$.

0.29. coNP is the class of problems for which we can reject proposed solutions in polynomial time.

0.30. $\mathrm{P} \subseteq \mathrm{NP} \cap \mathrm{coNP}$

0.31 (EXP)**.** $\mathrm{EXP} = \bigcup_{c\geq 1} \mathrm{DTIME}\left(2^{n^c}\right)$

0.32 (NEXP)**.** $\mathrm{NEXP} = \bigcup_{c\geq 1} \mathrm{NTIME}\left(2^{n^c}\right)$

Chapter 3: Diagonalization

0.33 (Main Idea)**.** For two complexity classes C_1 and C_2, we show $C_1 \subsetneq C_2$ by presenting L s.t. $L \in C_2$ but $L \notin C_1$.

0.34 (Time Hierarchy Theorem)**.** If f, g are time constructible function satisfying $f(n) \log f(n) = o(g(n))$, then we have $\mathrm{DTIME}(f(n)) \subsetneq \mathrm{DTIME}(g(n))$.

0.35 (Sketch for $\mathrm{DTIME}(n) \subsetneq \mathrm{DTIME}\left(n^{1.5}\right)$)**.** For the "diagonal" machine D: on input x run for $|x|^{1.4}$ steps on the universal TM $\mathcal{U}$ to simulate $M_x(x)$. If $\mathcal{U}$ outputs a bit $b \in \{0,1\}$, we yield the opposite, which is $1 - b$, and 0 if it fails to halt in time $|x|^{1.4}$. By definition, D always halts within $n^{1.4}$ steps and so any language decided by D is in $\mathrm{DTIME}\left(n^{1.5}\right)$. There is no machine M that always halts within n steps that can emulate D because D will have captured its result and yielded the opposite. Thus, $L \in \mathrm{DTIME}\left(n^{1.5}\right)$, but $L \notin \mathrm{DTIME}(n)$.

0.36 (Non-deterministic Time Hierarchy Theorem)**.** If f, g are time constructible funcitons satisfying the bound $f(n+1) = o(g(n))$, then $\text{NTIME}(f(n)) \subsetneq \text{NTIME}(g(n))$.

0.37 (Sketch for $\text{NTIME}(n) \subsetneq \text{NTIME}\left(n^{1.5}\right)$)**.** Apply lazy diagonalization. Define $f : \mathbb{N} \to \mathbb{N}$ as follows: $f(1) = 2$ and $f(i+1) = 2^{f(i)^{1.2}}$. We will find some n such that $f(i) < n \leq f(i+1)$. The goal for our diagonal machine is to flip the result on the set $\{1^n : f(i) < n \leq f(i+1)\}$. Define D as: If (1) $f(i) < n < f(i+1)$ then simulate M_i on input 1^{n+1} using non-determinism in $n^{1.1}$ time and output. Otherwise if (2) $n = f(i+1)$ accept 1^n iff M_i rejects $1^{f(i)+1}$ in $(f(i)+1)^{1.1}$ time. Thus, if $f(i) < n < f(i+1)$ then $D(1^n) = M_i\left(1^{n+1}\right)$, otherwise $D\left(1^{f(i+1)}\right) \neq M_i\left(1^{f(i)+1}\right)$.

0.38 (Ladner's Theorem)**.** Suppose that $\text{P} \neq \text{NP}$, then there exists a language $L \in \text{NP} \setminus \text{P}$ that is not NP-complete.

0.39 (Sketch for Ladner's Theorem)**.** We make some $L \in \text{NP} \setminus \text{P}$ "easy enough" using padding so that it is no longer NP complete, while keeping it "hard enough" so that it is not P. We do this by taking an NP complete language and "blowing holdes" in it.

Chapter 4: Space Complexity

0.40 (Space-bounded Computation)**.** Let $S : \mathbb{N} \to \mathbb{N}$ and $L \subseteq \{0,1\}^*$. We say that $L \in \text{SPACE}(s(n))$ if there is a constant c and a deterministic TM M that decides L with at most $c \cdot s(n)$ locations on M's work tapes (excluding the input tape) are ever visited by M's head during its computation on every input of length n. Likewise, say that $L \in \text{NSPACE}(s(n))$ if there is an NDTM M deciding L that never uses more than $c \cdot s(n)$ non-blank tape locations on length n inputs, regardless of its non-deterministic choices.

0.41. $\text{DTIME}(S(n)) \subseteq \text{SPACE}(S(n))$ since a TM can access only one tape cell per step, so a $\text{SPACE}(S(n))$ TM can run for much longer than $S(n)$ steps. In fact, halting behavior can run for as much as $2^{\Omega(S(n))}$ steps – as an example, a counter machine that counts from 1 to $2^{S(n)-1}$. Any language that is in $\text{SPACE}(S(n))$ is in $\text{DTIME}\left(2^{O(S(n))}\right)$.

0.42. For space constructible $S : \mathbb{N} \to \mathbb{N}$: $\text{DTIME}(S(n)) \subseteq \text{SPACE}(S(n)) \subseteq \text{NSPACE}(S(n)) \subseteq \text{DTIME}\left(2^{O(S(n))}\right)$.

0.43 (Configuration Graph)**.** A configuration of a TM M consists of the contents of all non-blank entries on M's tapes, along with its state and head position at a particular point in execution. For eveyr space $S(n)$ TM M and input $x \in \{0,1\}^*$, the configuration graph of M on nput x, denote $G_{M,x}$ is a directed graph whose nodes correspond to all possible configurations of M, where the input contains the value x and the work tapes have at most $S(|x|)$ non-blank cells. The graph has a directed edge from a configuration C to a configuration C' if C' can be reached from C in one step according to M's transition function. If M is deterministic, then the graph has out-degree one, and if M is non-deterministic, then the grpah has outdegree at most two. Let $G_{M,x}$ be the configuration graph of a space-$S(n)$ machine M on some input x s.t. $|x| = n$:

- Every vertex in $G_{M,x}$ can be described using $cS(n)$ bits for some constant c (depending on M's alphabet size and number of tapes) and in particular, $G_{M,x}$ has at most $2^{cS(n)}$ vertices.
- There is an $O(S(n))$-size CNF formula $\varphi_{M,x}$ such that for every two strings C and C', we have $\varphi_{M,x}(C, C') = 1$ iff C and C' encode two neighboring configurations in $G_{M,x}$. This can be expressed as an AND of many checks to see if the transition of the machine state is valid.

0.44 (Space Complexity Classes)**.** We can think of PSPACE and NPSPACE as analogs to time classes P and NP.

- $\text{PSPACE} = \bigcup_{c \geq 1} \text{SPACE}(n^c)$
- $\text{NPSPACE} = \bigcup_{c \geq 1} \text{NSPACE}(n^c)$
- $\text{LSPACE} = \text{SPACE}(\log n)$
- $\text{NLSPACE} = \text{NSPACE}(\log n)$

0.45 (Space Hierarchy Theorem)**.** If f, g are space constructible with $f(n) = o(g(n)) : \text{SPACE}(f(n)) \subsetneq \text{SPACE}(g(n))$.

0.46. We do not know if $\text{P} = \text{PSPACE}$, although we think no. Note that $\text{NP} \subseteq \text{PSPACE}$.

0.47 (PSPACE-completeness)**.** A language L' is PSPACE-hard if for every $L \in \text{PSPACE}$, $L \leq_p L'$. If in addition $L' \in \text{PSPACE}$, then L' is PSPACE-complete.

0.48 (SPACETMSAT)**.** $\text{SPACETMSAT} = \{(M, w, 1^n) : \text{DTM} M \text{ accepts } w \text{ in space } n\}$.

0.49 (Quantified Boolean Formula)**.** A QBF is of form $Q_1x_1.Q_2x_2.\dots.Q_nx_n.\varphi(x_1, x_2, \dots, x_n)$ where each $Q_i \in \{\exists, \forall\}$, x_i range over $\{0, 1\}$, and φ is a plain, unquantified formula.

0.50. Since all variables of a QBF are bounded by some quantifier, QBF is always either 1 or 0 as a result.

0.51. (TQBF) TQBF is PSPACE complete.

0.52 (Sketch for TQBF $\in$ PSPACE)**.** Let n be the number of quantified variables in TQBF Ψ, and proceed by induction. If $n = 0$, then we are done. Otherwise for $n > 0$, the formula will have form $Q_nx_n.Q_{n-1}x_{n-1}.\dots.Q_1x_1.\varphi$. If Q is $\exists$, we check if either $\Psi|_{x_n=1}$ or $\Psi_{x_n=0}$ is true. If Q is $\forall$, we check if both $\Psi|_{x_n=1}$ and $\Psi_{x_n=1}$ are true.

0.53 (Sketch for TQBF is PSPACE-hard)**.** For some M that can decide L in $S(n)$ space, we can encode the configuration graph as a TQBF Ψ such that on input $x \in \{0, 1\}^n$, $\Psi \in \text{TQBF} \iff M(x) = 1$. Observe that because M can decide L in $S(n)$ space, we thus need $S(n)$ bits of information to encode M's configuration graph on input x We use the property that every configuration graph for M on input x has a formula $\varphi_{M,x}$ such that for two strings, $C, C' \in \{0, 1\}^m$, we have $\varphi_M(C, C') = 1$ iff C and C' encode two adjacent configurations in the configuration graph. When we plug in C_{start} and C_{accept}, this gets us what we want. Define Ψ by induction.
Naive solution: let $\Psi_i(C, C') = 1$ iff there is a path of length at most 2^i from C to C' in $G_{M,x}$. Define the formula $\Psi_i(C, C') = \exists C'' \Psi_{i-1}(C, C'') \land \psi_{i-1}(C'', C')$. However this is a problem because Ψ_i may double each layer.
Less naive: $\Psi_i(C, C') = \exists C''.\forall D_1.\forall D_2.\, ((D_1 = C \land D_2 = C'') \lor (D_1 = C'' \land D_2 = C')) \implies \psi_{i-1}(D_1, D_2)$.

0.54. TQBF $\in$ NPSPACE because we have said nothing about the out-degrees of the configuration graph above.

0.55 (Savitch's Theorem)**.** For any space constructible $S : \mathbb{N} \to \mathbb{N}$ with $S(n) \geq \log n$, $\text{NSPACE}(S(n)) \subset \text{SPACE}(S(n)^2)$.

0.56 (Sketch for Savitch's Theorem)**.** The configuration graph of some M that works in NSPACE($S(n)$) has at most $2^{O(S(n))}$ vertices. This can then be encoded as a $O(S(n)^2)$ space formula for checking.

Chapter 5: The Polynomial Hierarchy and Alterations

Chapter 6: Boolean Circuits

Examples